Digit Exp Math Educ
DOI 10.1007/s40751-017-0028-x

MATHEMATICS AND PROGRAMMING

Bridging Primary Programming and Mathematics: Some Findings of Design Research in England

Laura Benton[1] · **Celia Hoyles**[1] · **Ivan Kalas**[1,2] · **Richard Noss**[1]

Abstract In this paper we present the background, aims and methodology of the ScratchMaths (SM) project, which has designed curriculum materials and professional development (PD) to support mathematical learning through programming for pupils aged between 9 and 11 years. The project was framed by the particular context of computing in the English education system alongside the long history of research and development in programming and mathematics. In this paper, we present a "framework for action" (diSessa and Cobb, *Journal of the Learning Sciences, 13*, 77–103, 2004) following design research that looked to develop an evidence-based curriculum intervention around carefully chosen mathematical and computational concepts. As a first step in teasing out factors for successful implementation and addressing any gap between our design intentions and teacher delivery, we focus on two key foundational concepts within the SM curriculum: the concept of algorithm and of 360° total turn. We found that our intervention as a whole enabled teachers with different backgrounds and

The research reported in this paper forms part of a larger project, the ScratchMaths project 2014–17 funded by the Educational Endowment Foundation. The project is being independently evaluated following a randomized control trial model involving over 100 primary schools across England.

✉ Laura Benton
l.benton@ucl.ac.uk

Celia Hoyles
c.hoyles@ucl.ac.uk

Ivan Kalas
i.kalas@ucl.ac.uk; kalas@fmph.uniba.sk

Richard Noss
r.noss@ucl.ac.uk

[1] UCL Knowledge Lab, UCL Institute of Education, University College London, 23-29 Emerald Street, London WC1N 3QS, UK

[2] Department of Informatics Education, Comenius University, Bratislava, Slovakia

Published online: 23 January 2017

levels of confidence to tailor the delivery of the SM in ways that can make these challenging concepts more accessible for both themselves and their pupils.

Keywords Children's programming · Computing curriculum · Mathematics · Scratch · Design research

Background

In recent years the teaching of computer programming to children has become widespread. There has been an enhanced interest in introducing programming in school in many countries around the world, both informally as part of after-school 'coding' clubs and formally through the school curriculum (Sentance and Csizmadia 2015). England is one of only a few countries that has made the teaching of programming a compulsory part of primary education to date, but many other countries are currently considering changes to their education systems (Passey 2016). From September 2014 all primary schools in England were required to teach the new national computing curriculum, which includes learning about how computational systems work, using technology to develop ideas as well as designing and building their own programs. Many primary teachers (who are typically generalists) are unlikely to have the appropriate skillset to teach this new technical subject. Therefore there is a need for widespread professional development (PD) to support teachers in gaining the necessary experience, technical skills, confidence and understanding of suitable pedagogies in order to implement this new curriculum successfully (Brown et al. 2014; Sentance and Csizmadia, A 2016). And of course, there is an equally significant challenge to reassess what can be learned and taught.

The teaching of computer programming in English schools is not new, with educational programming languages such as Logo and BASIC widely used in both primary and secondary education settings during the 1980s and 90s. However, Manches and Plowman (2015) highlight that more recent discussion around how to teach programming in schools has often omitted the earlier research conducted during this time: there is, for example, a largely overlooked series of conferences and proceedings held throughout the early 1980s: *Logo and Mathematics Education (LME)* some of which are captured in Hoyles and Noss (1992). During the late 1990s programming slowly faded from the curriculum as it became subsumed and eventually replaced entirely by the subject of ICT (Information and Communications Technology), which focused more on the *use* of technology than on its creation (Brown et al. 2014).

Research on the efficacy of programming has produced mixed results (Clements 1999; Voogt et al. 2015). From the point of view of learning programming per se, some of the key challenges within early research were difficulties with programming syntax, dealing with error messages along with the severely limited access to technology within the classroom (Resnick et al. 2009; Lewis 2010). Since then there have been significant developments in novice programming languages which have overcome some of these issues, especially in the fashionable swing towards graphical rather than text-based programming. In addition, technology access has become increasing commonplace within schools with resources readily available for sharing through the Web. Much can be done now to redress the lack of clarity in earlier work concerning computational

thinking, the role programming plays in its development, and in particular a clarification of what other knowledge might become learnable with programming, rather than focusing only on programming for its own sake. Our interest focuses on the relationship between learning to program and learning to express mathematical ideas through programming.

The ScratchMaths (SM) project thus set out to explore this potential for the 9–11 primary[1] age group (upper Key Stage 2 (KS2)) in light of the recent curriculum changes and the renewed enthusiasm and motivation for the teaching of programming in schools. The project consists of a design phase following which the intervention is implemented over 2 years with the same pupils (see Table 1). The project evaluation has two components:

> Part 1 is undertaken by an independent evaluation team from another university. It consists of a quantitative evaluation following a randomised control trial model using the scores on the national mathematics assessment (termed KS2 test) taken by all pupils in England at age 10–11 years;
> Part 2 is a qualitative evaluation to identify potential causal mechanisms for the quantitative outcomes, led by our team, a sub-part of which is focused upon within this paper to enable key components of the intervention to be explored in depth.

History of Programming and Mathematics Learning for Children

During the 1970s and 80s research interests gradually focused on learning mathematics through writing algorithms and the potential framework for learning mathematics that Logo programming could provide (Noss and Hoyles 1996). Early research also highlighted the potential of expressing mathematical ideas as computer programs (Feurzeig et al. 1969; Hatfield and Kieren 1972). Noss and Hoyles (1996, pg. 55) argue that writing a computer program "provides a broad canvas on which the learner can sketch half-understood ideas, and assemble on the screen a semi-concrete image of the mathematical structures he or she is building intellectually". Many but not all of such studies followed a constructionist paradigm, in which learning is viewed as building knowledge structures and happens most effectively when the learner is actively engaged in "constructing a public entity" (Papert and Harel 1991). Following this paradigm, Papert (1980, 1993), for instance, illustrated how suitably designed Logo experiences could stimulate children's cognitive development and offer them the opportunity to explore "powerful ideas" related to mathematics. He cited many potential benefits of Logo programming from the learning of mathematics which included specific concepts such as angles, degrees and variables, as well as more general problem-solving strategies like decomposition and debugging (Miller et al. 1988). Clements and Sarama (1997) also highlighted the opportunities Logo could provide children to manipulate concrete instances of mathematical ideas, to facilitate connections between concrete experiences and more abstract mathematics as well as allowing children to use mathematics in ways that are more personally meaningful to them.

[1] Primary education in England includes pupils aged 5–11 years.

Table 1 Overview of ScratchMaths project phases

Year of project	Project activity	Evaluation components	
		Qualitative	Quantitative
1	• Design of intervention	Design school visits	
2	• SM intervention implemented with Year 5 (age 9–10) pupils • Computational thinking (CT) test administered at end of year	Subset of trial school visits	Analysis of Computational Thinking test results
2	• SM intervention implemented with *same* pupils (now Year 6 - age 10–11) • KS2 Mathematics test	Subset of trial school visits	Analysis of KS2 mathematics results

However, findings from research into the 'impact' of computer programming on children's learning of mathematics has been inconclusive. The diversity of research paradigms, experimental conditions and disappointingly poor measures of 'transfer' makes it difficult to compare. In addition, asking for the impact of programming ignores the complexity of the issues at stake, and rich interplay of context that impact is likely to miss. Furthermore much of this early research focused on out-of-school or selective settings rather than typical classroom contexts, which would necessitate more consideration of curriculum and teachers (Lye and Koh 2014). In light of concerns about the applicability of the research results to the wider population, many researchers in this area employed design research methods, requiring a clear theoretical basis from which the outcomes could be generalizable (Prediger et al. 2015). In addition, in contrast to much of this early research, a key goal of the SM project was that the intervention would be accessible to all pupils regardless of background, gender or attainment level.

Researchers have attempted to provide explanations for the variation in outcomes in diverse studies. These include the potential impact that the children's existing knowledge and abilities as well as stage of development may have, with Clements (1985, pg. 59) claiming that in order to learn Logo "certain mathematical, spatial and problem-solving abilities are required" and any potential effects take time to emerge (Clements and Sarama 1997). Furthermore Clements and Sarama (1997) propose the need for specially-designed and sequenced activities, with Hoyles and Noss (1992) raising concerns about children bypassing mathematical ideas within less structured learning activities and without teacher guidance (see also Clements 1999). Crucially Salomon and Perkins (1989) suggest that the Logo programming projects that demonstrated more positive 'transfer' effects promoted mindful abstraction of learning outcomes. This relates, of course, to the role of the teacher and many researchers highlight that this is critical not least to make explicit and systematic links to pupil's existing mathematical knowledge and experiences (Clements 1985, 1999; Clements and Sarama 1997) as well as in facilitating their pupils' discovery and understanding of the powerful ideas they encounter within a programming environment such as Logo (Yelland 1995; McCoy 1996).

Many new programming environments have been developed since much of the research in programming and mathematical learning was undertaken: some of which is enjoying reasonably widespread adoption (Duncan et al. 2014). Some have been specifically developed for novices and young people, several of which are blocks-

based (Duncan et al. 2014; Weintrop and Wilensky 2015). One of the most popular of these is Scratch (used by millions of children worldwide, most commonly in out-of-school contexts), developed at MIT and a descendent of Logo (Resnick et al. 2009).[2] Resnick et al. (2009) suggest that Scratch provides opportunities for learning important mathematical and computational concepts, as well as offering space for creative thinking, systematic reasoning and collaborative work.

Given its mass appeal, ease of access and suitability for children, Scratch was chosen as the programming environment to be used in this study. We also took from our review of prior research the need to take seriously, first the design of a curriculum (rather than isolated activities) and second, the teacher's role and how teachers can be supported.

There has been a significant investment in supporting the introduction of the new statutory computing curriculum within English schools.[3] In our view there remains an urgent need to support this initiative to make connections with other subjects for the benefit of both. So for example to research how programming can be exploited as a modelling tool across the curriculum and to identify ways to show the effectiveness of this approach. Our project takes this forward in the context of programming and mathematics and seeks to build on previous research as well as take advantage of the new programming languages and widespread connectivity.

Aims and Structure of the Research

The 2-year SM intervention includes the provision of detailed curriculum materials (for 20+ lessons in each year) as well as PD (2 days each year) both of which are underpinned by an explicit pedagogical "framework for action" (diSessa and Cobb 2004) developed through a process of design research. In the first year, for Year 5 (Y5) pupils, computational concepts are foregrounded with mathematical components implicit in the approach. In the second year, the same pupils now in Year 6 (Y6) are introduced to mathematical concepts and mathematical reasoning explicitly through programming with links specified to the (statutory) primary mathematics and computing curricula. This enables parts of computing to be taught within or as a supplement to mathematics lessons, which are prioritized by all schools as a key area of assessment. Pupils will be developing their programming and mathematical understandings over this period, thus our first aim is to establish:

> What are the characteristics of an evidence-based curriculum intervention in both mathematics and programming for primary school pupils (aged 9–11 years)?

In their classification of theoretical methodologies, diSessa and Cobb (2004) propose that "frameworks for action" incorporate a prescription of the pedagogical strategies within the intervention design, that specify goals and actions in instruction. They go on to highlight a common difficulty many frameworks for action experience

[2] Scratch is freely available for use both online and offline, and is intended to encourage people of diverse ages, backgrounds and interests to engage in creating and sharing interactive projects, such as stories, games, animations or simulations (Resnick et al. 2009).

[3] for example the substantial Government funded Computing at School (CAS) Programme across England with its 150 regional hubs through which nearly 4000 teaching resources are shared.

in achieving their intended goals in practice, which they term "managing the gap". This gap can include atheoretical aspects of design such as technological infrastructure, the nature of the classroom discourse and practical aspects such as time pressure in schools. diSessa and Cobb (2004 pg. 82) states that ideally "pedagogical strategies and conjectures are separated by a *carefully considered and articulated gap* from the theory or theories that explain or motivate them". Therefore our second aim is to establish:

> What is the gap between the intervention designers' intentions and the teachers' implementation of the intervention in practice for specific computational and mathematical concepts and how does this impact pupil understanding of the concepts involved?

Table 2 Overview of SM design research phases, activities and outcomes

Phase	Design research activity	Outcome
Preliminary research phase	Needs and context analysis – co-design workshops with teachers and lesson observations	Tentative framework for action – the "5Es": see below
	Literature review and expert appraisal – of previous related research and existing resources and schemes of work	
Prototyping phase	Specifying overall description of the SM intervention guided by the tentative framework for action	Design proposal
	Designing high-level scheme of work including module structure, key concepts and operations to be addressed in each year	Global design (including specified framework for action)
	Building a fully specified subset of the activity content and support materials to be trialled in the classroom	Partly detailed product (including refined framework for action)
	Iterative formative evaluation of activity content and support materials in a range of settings to establish practicality and potential effectiveness of intervention (in design school classrooms)	Completed product (including final framework for action)
Summative evaluation phase	Trial of completed product with different teachers in the same school	Identification of factors impacting the 'gap' between intentions and implementation

Design Research Phases

Design research necessarily comprises multiple cycles, which involve a number of different design and research activities. Nieveen and Folmer (2013) divide these activities into three distinct phases: (i) preliminary research phase; (ii) prototyping or development phase; and (iii) summative evaluation phase. Table 2 presents an overview of these three phases and the activities conducted during these phases along with their outcomes in the design of the SM intervention. Details and explanation of terms will be provided below. We describe the design research process undertaken on the SM project in terms of these three phases.

Preliminary Research Phase

The aim of the preliminary research phases is to identify the educational problems the intervention needs to address through first establishing the current educational context and issues, and then proposing tentative intervention features that would fit into this context and address the relevant issues (Nieveen and Folmer 2013). Two key activities undertaken during this phase include an analysis of the user practice (needs and context analysis) and an exploration of the scientific knowledge base (literature review and expert appraisal).

Needs and Context Analysis To establish the needs of primary school teachers and their pupils (our stakeholders) we undertook a series of co-design workshops with a number of teachers from different schools. Four London state primary schools were recruited to act as 'design schools' in the initial design phase during year one of the project. One or two teachers responsible for teaching computing to the target age range from each school (total seven teachers) participated. The majority of these teachers were also class teachers and had a range of different experiences with Scratch. All teachers participated in five co-design workshops, which were intended to provide an opportunity to build collaborative working relationships with the project team as well as sharing their experience of teaching computing and mathematics. This allowed us to establish the teachers' perceptions of the current situation in primary schools and how this could be best addressed through our intervention. These workshops also provided a guide to the appropriateness of different activities, with several activities being acknowledged as too difficult for use in primary practice, while others were taken forward as a basis for the final activities.

Members of the project team also visited each school to observe computing and mathematics lessons. The resulting field notes from both the workshops and school visits aided the elaboration of the current context, the key needs of the teachers and pupils, and any specific issues that needed to be addressed.

Literature Review and Expert Appraisal In conjunction with the needs and context analysis a review of relevant literature, both academic literature, particularly focusing on the extensive work that had previously been undertaken by members of the project team and others in teaching Logo within schools, and a range of existing computing textbooks/schemes of work aimed at supporting teachers in teaching programming as well as mathematics textbooks for primary teachers, were examined. The aim of this work was to establish what were the characteristics of previously successful (but

smaller scale) interventions and also to identify content that could inspire specific learning activities.

Outcomes of Preliminary Research

The key outcome from this initial phase of the design research process was the development of an explicit framework for action to underpin pedagogic strategies of implementation. We specifically chose to focus on developing a framework for action as it guides us in finding the right level at which to intervene. This framework arose from our aim to design a constructionist approach to learning (as mentioned earlier), and was further informed by the findings from the design workshops and observations, as well as our review of the existing programming schemes of work. We called this framework the "5Es" (see Benton et al. 2016 for detail), which will in turn guide the intervention structure and activity design as well as the design of the PD. It is important to note that these constructs are unordered and intended to be flexible in their application within different contexts.

Explore Opportunities should be provided for pupils to investigate ideas by trying things themselves and debugging errors. Constructionist approaches value learning by exploring the thinking processes engendered when using computers, thus helping to understand what can or cannot be done (Papert 1993). However, during the co-design workshops after undertaking some of the Scratch activities for themselves several of the teachers highlighted a need for the material to handle progression more incrementally to reduce the likelihood of pupils becoming disconnected from the task. In other words, we found it necessary to consider an appropriate balance of structure and space for individual exploration. Many teachers highlighted the challenge of fitting this new intervention into an already crowded timetable. Some of the existing Scratch-based schemes of work try to cover the entirety of the Scratch functionality as well as making large jumps in complexity, but in this context it is important that the exploration is focused on introducing the key programming concepts (guided by the primary computing curriculum) which incrementally build on each previous step and does not succumb to a technocentric approach focused on learning to use the programming environment rather than learning through it (Brennan 2015).

Envisage Pupils should be encouraged to predict outcomes before running scripts and then reflect on the actual outcome. A key approach to understand the power of computer programming is in reflection on and re-evaluation of intuitive expectations and knowledge, so these need to be made explicit in some form before feedback is obtained (Papert 1993). One approach which was encouraged in relation to envisaging in Logo was the use of "body syntonic" reasoning where the programmer imagines themselves as the object they are programming with (Watt 1998). During the lesson observations a weekly mathematics lesson was observed that was specifically focused on problem solving, during this lesson the teacher highlighted the importance of understanding the problem solving process or strategy used rather than focusing just on the solution. The development of mathematical problem solving and reasoning skills are a key goal of the primary mathematics curriculum and activities which encourage prediction can help to support this goal.

Explain Opportunities should be provided for whole-class discussions led by teachers as well as with peers through the inclusion of reflective questioning. Harel and Papert (1990) highlight the cognitive benefit of generating verbal explanations, which helps to clarify ideas. The process of reflection and explicit articulation required to generate these explanations is a key part of the constructionism approach (Noss and Hoyles 1996; Brennan 2015). During the co-design workshops some of the teachers highlighted a need for more discussion to support understanding and develop problem-solving strategies. The observations of computing lessons where pupils were using Scratch also raised a potential concern around the extent to which all pupils understood the control structures they were using, which suggests a need to explicitly build and verify this understanding through reflective questioning and discussions.

Exchange Meaningful opportunities to share and build on others' ideas should be included. The teachers in the co-design workshops described the big differences among their pupils in terms of their Scratch experience, support needs and confidence, and also expressed a desire for higher attainers. Higher attaining pupils providing peer support to classmates may be an opportunity to address these issues. This is a particularly key focus as little previous research has focused on designing and delivering similar interventions for a diverse group of pupils, such as those found in a typical primary classroom (Israel et al. 2015).

bridgE The links with the primary mathematics curriculum as a powerful idea should be made explicit. Ideas are viewed as powerful partly through their connection to other disciplines, here to mathematics (Papert 1993). The teachers in the co-design workshops wanted these connections to made clear within the context of the existing primary curriculum, with the development of the content for the first year being led by fulfilment of the programming aspects of the computing curriculum and subsequently identifying the mathematical connections.

Prototyping Phase

During the 'prototyping phase' we followed an iterative design approach, which involves continually refining and evolving the intervention to reach a final version. Nieveen and Folmer (2013) suggest beginning with the development and evaluation of a small part of the intervention, such as a complete lesson, and then apply the findings from this (in terms of shortcomings and successes) to subsequent parts of the intervention. At this stage the evaluation is seen as formative, and it is a crucial element of the prototyping phase to enable issues to be identified and potential improvements to be generated.

Nieveen and Folmer (2013) propose four stages within the prototyping phase:

- *Design proposal:* a general description of the SM intervention, which was elaborated on through the preliminary research phase by the development of the 5Es teaching framework (i.e. the framework for action).
- *Global design:* after the preliminary design phase a global intervention design was developed, which initially included an overview of the modules of work for the first year (i.e. Y5) of the SM intervention (to which an overview of Y6 was later added,

see Table 3) and a high-level description of the key concepts they would each address (see Table 4 for an example subset of the global design for the concept of algorithm).
- *Partly detailed product:* the first few activities were fully specified to enable them to be evaluated in school with a teacher and pupils (i.e. formative evaluation). This stage is described in more detail in the following section.
- *Completed product:* at the end of the prototyping phase the intervention should reach a completed state, which is suitable for use in the intended scenario.

At different stages of the formative evaluation different quality criteria are applicable, which include: relevancy (or validity); consistency (logical design); practicality and effectiveness (Nieveen and Folmer 2013). Relevancy and consistency are the main focus of the earlier stages, i.e. design proposal and global design, and help to ensure the intervention is aligned with the national curriculum as well as standard primary education practice. These early stages of the formative evaluation involved members of our interdisciplinary project team, who have extensive experience in both mathematics and computing education research as well as undertaking PD with teachers, working together to identify links between the mathematics and computing curriculums. After the intervention had been further elaborated the practicality of the design and potential effectiveness was evaluated in context, and the boundaries of the invariant features of different implementation. This process enabled us to establish the "boundaries of immutable features of the intervention" (Hung et al. 2010), which enable the identification of any lethal mutations of the intervention (i.e. changes made in the implementation of the intervention which contradict the proposed framework) during the summative evaluation phase.

All four of our design schools participated in the formative evaluation of the Y5 materials, with one school (School A) trialling all the content over the course of a year and the other three schools (Schools B, C and D) testing selected activities. The teachers had all attended our earlier co-design workshops. One to three project researchers undertook observations of the lessons and wrote extensive field notes, conducted informal conversations with both the teacher and pupils as well as collected all pupil work that was produced during the lesson (i.e. Scratch projects and any completed paper-based worksheets). After each lesson the researchers met together to discuss what happened during the lesson and identify any potential areas for improvement, triangulating the collected data where necessary. The materials were then refined and re-tested with either the same class (where substantial changes were made to the core focus of the activity) or a different class in either the same school or another school (where more minor improvements were made). The iterations moved from focusing on the practicality of the activity design for the primary classroom to the appropriateness of the activity content for pupils in the target age range.

Outcomes of Prototyping Phase

Here we focus upon two distinct threads, which resulted from the prototyping phase – the curriculum design, which specifies aspects of the specific content of the learning activities and the PD, which identifies implications of the accompanying programme of

Table 3 Overview of Y5 and Y6 ScratchMaths modules, highlighting the approach to the concepts of algorithm and 360° total turn as in specific modules (in *italics*)

Module	Computing concepts/operations	Mathematics concepts
1: Tiling Patterns • Stamping different patterns e.g. repeated linear or circular, symmetrical and alternating • Defining own pattern block to use in creating more complex patterns	Introduces the key computational concepts of sequencing, repetition, algorithm, debugging and abstraction through the use of definitions. *Multiple pattern stamping algorithms are used to explore ordering and pattern creation process.*	Links to symmetry, angles, 360° total turn and negative numbers through building patterns. *Majority of patterns are based around using notion of 360° total turn.*
2: Beetle Geometry • Drawing roman numerals • Drawing different regular polygons • Drawing dots and dotted patterns • Combining elements to draw nature scenes	Introduces initialisation, randomness and expressions as well as consolidating earlier concepts from Module 1. *Algorithms are explored through comparing the outcomes of different sequences of the same instructions and physically instructing and stepping through each command.*	Links to creating different regular polygons. *360° total turn is used to create different regular polygons.*
3: Interacting Sprites • Building isolated behaviours for multiple sprites e.g. walking, jumping • Building interactions between sprites • Building complex behaviours with multiple interactions to tell a story	Introduces conditions, broadcasting and more expressions, thus allowing interactions between multiple actors and parallel behaviours. *Different algorithms are explored through replacing one change of the sprite state by a sequence of smaller changes e.g. jumping in one go vs. repeating a sequence of small jumps*	Links to coordinates, multiplication and factors.
4: Building with Numbers • Building a place value model up to four places • Converting place value model to represent time • Exploring the place value model for conversions	Uses broadcasting to build models with synchronised parallel behaviours in several mathematical contexts.	Links to place value as well as conversions of length, weight and time (including connections between analogue and digital representations).
5: Exploring Mathematical Relationships • Drawing random polygon night sky scenes • Drawing mathematically similar rectangles	Introduces the concept of variable. *The effect of the order of commands in an algorithm is explored through the need to store multiple values (e.g. the number of sides and length of side to be drawn).*	Links to different types of mathematical relationship including proportionality and ratio in the context of drawing rectangles. *The invariant relationship between 360° total turn, number of sides and angle of turn is explored through drawing polygons.*
6: Coordinates and Geometry • Drawing filled rectangular areas composed of random dots within a defined area of the coordinate grid • Transforming shapes and patterns	Explores the control of constrained randomness through number of repetitions and defined ranges.	Links to the translations and reflections of regular polygons through the coordinates system

teacher development. (There is a further thread not explicitly addressed here - the refined framework for action, which attempts to address the gap between theory and pedagogy and is discussed in more detail within Benton et al. (2016)).

Curriculum Design and Content The global curriculum design was initially developed through a series of project team meetings, which were intended to identify the key computing concepts for the Y5 curriculum and make links between this and the existing primary mathematics curriculum. Table 4 shows the part of the design for the first module of the Y5 curriculum for the concept of algorithm, which focused on stamping patterns.

The global design of the intervention set out the introduction, integration and development of the key mathematical and computational concepts addressed with the SM curriculum. In this paper we focus on the concept of *algorithm* and on the concept of *360° total turn.* We developed the SM materials with these two concepts as rich sources of mathematical ideas, which we sought to exploit through expression in Scratch. These two concepts are revisited in different contexts throughout the SM curriculum (see Table 3) and are also central to both the primary computing and mathematics curriculums. It is therefore important that pupils are able to grasp these concepts early on because later activities aim to build on this foundational knowledge.

Algorithm – an unambiguous description of how to solve a task in a finite time – is one of the primary concepts of the new computing curriculum and is introduced in Key Stage 1 (ages 5–7). We had noted in our review of curriculum materials that common introductory examples of algorithms within the primary curriculum usually comprised describing everyday or school activities like making a sandwich or multiplying two numbers. These examples only illustrate a limited number of the characteristics of algorithm, namely order and fail to address the importance of precise language. Furthermore given our constructionist approach we also wanted to introduce the concept of algorithm where the strategy would not be trivial, not known in advance, not unique (i.e. meaningful to consider and compare alternative approaches), could be applied in other (and sometimes unexpected) contexts and often generalizable to a broader set of tasks.

In SM the concept is introduced in Module 1: Tiling Patterns (Moving Forwards and Backwards) through exploring two different stamping strategies to produce a pattern using a small set of blocks: the first algorithm creates a circular pattern by repeating the

Table 4 Example global design for the concept of Algorithm from Module 1 (Tiling Patterns) on stamping patterns

Computing concepts and operations	Scratch constructs and operations	Links with primary mathematics curriculum concepts and operations
Algorithm and program • Build • Run • Use bounded loop • Edit, modify	Script • Create it by snapping blocks together • Run it by clicking it (a script with or without a hat block) • Modify it, duplicate, delete • Use repeat as a basic control structure • Use wait block to slow down the execution	Y5: Geometry – properties of shapes • Angles • 360° total turn

move – turn – stamp steps, while the second one creates a pattern starting from its centre of a circle and by repeating the *move – stamp – move backwards – turn* steps.

The strength of the algorithm derives from the fact it can be repeated with different tiles (costumes), repeated several times with multiple tiles, or generalised by replacing basic stamping by running a pupil-defined command (new block) - (Fig. 1 illustrates these outcomes for algorithm two).

In addition, a later activity involves the application of algorithm two in a different context, drawing the Earth covered by random trees and houses (see Fig. 2).

The concept of a *360° total turn* is a key mathematical idea with deep connections to several computational concepts that re-appear throughout the whole SM intervention: the first algorithm is introduced through repeatedly clicking a script of three blocks an approach that gives a visual interpretation of the idea of algorithm, motivates the counting of the number of clicks to complete the whole pattern and naturally introduces the repeat control structure (some outcomes are shown in Fig. 3). In this way it provides a preliminary context in which to *discover the connection* between the angle of turning and the repeat number. Later (in Module 2: Beetle Geometry) the same approach is used to draw a line and turn – again and again, leading to drawing a regular polygon. Following on from this, pupils explore an alternative *knowledge-led* approach where they are expected to use their understanding of this connection to build a script that draws a regular polygon, a square, a triangle and then a general polygon.

Here we describe in more detail the foundational activity from Module 1 – *Moving Forwards and Backwards* – which introduces the concept of algorithm and at the same time connects with the concept of a *360° total turn*. We refer to this later as the *research activity.* The first algorithm had already been introduced and this activity sets the pupils the task to build a rose pattern (see Fig. 4) using the second algorithm. Pupils have not only to build the new algorithm but also combine it with the concept of a *360° total turn* to create a complete circular pattern. The key objectives pupils need to achieve in order to successfully complete the activity are:

Objective 1 To build a script that follows the new algorithm: *move – stamp – move backwards – turn*.

Objective 2 To use mathematical knowledge of a full 360° total turn to stamp a circular pattern with no overlapping stamps.

Objective 3 To use mathematical knowledge of positive and negative numbers to move back to the centre of the pattern after each stamp (not addressed explicitly in this paper).

The global design of each module served to structure the development of learning activities, which addressed computational and mathematical concepts (as illustrated above) through the strategies specified within the 5Es framework. Each activity was

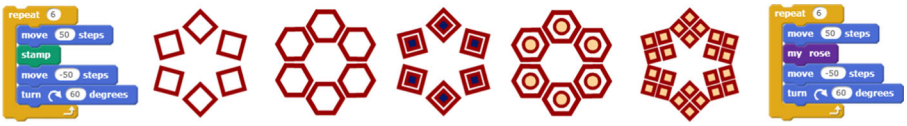

Fig. 1 Replacing basic stamping (Script on Left) with a user-defined block – My Rose (Script on Right)

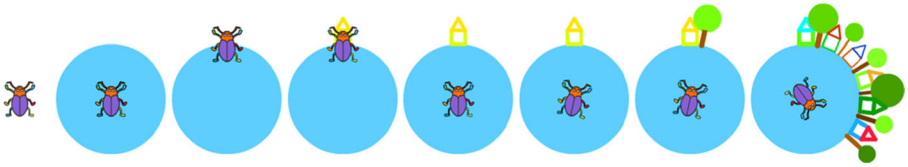

Fig. 2 Applying generalised versions of the two algorithms in a new drawing context to create a complex design

tested with a minimum of three classes of Y5 pupils (in School A), with selected activities being tested in up to two further schools (from Schools B, C and D). Below we describe some of the findings related to the implementation of the intervention, which addressed the two key concepts described above and the implications these had for the design of the PD programme for teachers.

Professional Development (PD) To support the curriculum implementation, a PD programme of 2-days per year was designed to support teachers in delivering the content of the SM intervention. Below we discuss some of the findings from the formative evaluation in terms of our two key computational and mathematical concepts and the resulting implications for the PD design.

Concept of algorithm: Several of the teachers from the design schools found it challenging themselves to *envisage* the outcomes of scripts for the two algorithms and also to *explain* them to pupils: they struggled to explain the difference between the two. This highlighted a need during the PD for developing strategies and tools to support the process of envisaging (for the teachers themselves) as well as providing them with opportunities as to how to best use these strategies and tools in the classroom (e.g. see Fig. 5). The developed strategies and tools included:

- The use of physical demonstrations and pupil walk-throughs of scripts
- Using the wait block within Scratch to slow down the running of the script
- Support videos that can be shown during class that highlight the differences between scripts
- Paper-based tools e.g. cut-outs of the sprite, use of wheeled toys.

The teachers were shown these strategies and tools during the subsequent PD sessions and were given the opportunity to try some of them out for themselves during the practical activities.

360° total turn: it was clear during the formative evaluation in the design schools that the teacher had an important role in highlighting this concept and making the connections between the algorithm and the mathematical structure. Pupils rarely

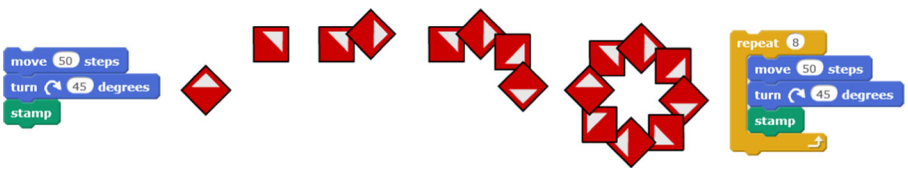

Fig. 3 Clicking three isolated blocks repeatedly… to discover the idea of 360°

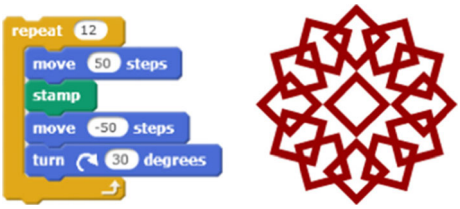

Fig. 4 Example rose pattern (Script and Pattern Outcome)

'discovered' this through their own exploration, particularly as they could put a larger number within the repeat block and this would still make a visually complete circular pattern.

When pupils were asked to articulate their strategies for calculating the values in the move and turn blocks to re-create different 360° circular patterns, we identified three types of response to discuss in the PD – to ensure teachers were aware of them and could develop appropriate support for children using them. These strategies included:

- *Guessing or using incorrect calculations*: One pupil stated that she had simply guessed the values. Other pupils explained flawed strategies related to counting the number of Tile stamps but with a lack of understanding about the behaviour of the sprite in Scratch "*12 because…it will go around twice because it might do half and then it might not do the other bit so to make sure it goes around twice*", or attempting to undertake some form of calculation not related to any knowledge about 360° "*20 because 30 minus 30 is 60 and 60 is the highest, 30 is the second high and I think −30 is the third and if I do 20 I think it's the lowest*".
- *Counting and estimation*: some pupils counted the number of stamps on the pattern. They also estimated based on their existing knowledge of angles using the picture of the pattern e.g. "*So it would be 90° because if you got a tile there and then I know that it turns that way for 90° and that's the same here*" and "*Well you can look at*

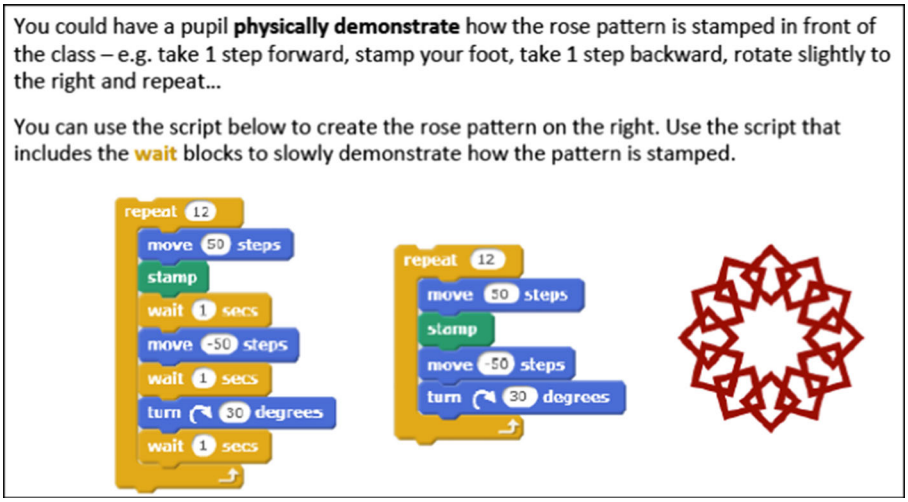

Fig. 5 Example of support strategies included within the teacher materials

the sprite and see how the first one is which is at the top in the middle and the sprite has turned right it looks like it's turned halfway which would probably by 45… because 45 is half of 90 and 90 is like the main degrees you can turn".

- *Correct calculation*: other pupils were able to explain the calculations they needed to use "*6 times 60 equals 360*" or "*you would divide 360 by 8*" or to find out what goes into 360 eight times. However the pupils were often unable to calculate these values in their head and therefore even though they may understand the concept this would not be reflected within their script and would require deeper probing from the teacher.

The refined curriculum content and PD programme was subsequently delivered to over 100 teachers at the start of the summative evaluation phase, which is discussed within the following section.

Summative Evaluation Phase

For the first stage of the qualitative summative evaluation we focused on the *research activity* described above.

The same research activity was observed in two Y5 classes in a single trial school (School E - note all trial schools are expected to deliver at least the core content from each of the six SM modules). The school was a large primary in London with an above-average number of pupils with special educational needs and from low-income families. In Class 1 there were 26 pupils and in Class 2 there were 29 pupils. All pupils worked individually on their own laptop computers (although partway through the lesson some pupils had to pair up due to technology problems). A researcher observed each class, which each lasted 45 minutes, and took detailed field notes.

Teacher 1 had 3 years of teaching experience, had studied mathematics up to GCSE level (aged 16) and had been using Scratch regularly with her class. During the PD she exhibited much lower confidence with Scratch than Teacher 2. Teacher 2 had 5 years of teaching experience, had studied mathematics up to A-Level (aged 18) and had been using Scratch regularly with her class and in after-school clubs. The teachers used the same internal mathematics assessment system assigning pupils a level based on periodic testing: the level ranged from 1a (lowest) to 5c (highest). On average, the level of mathematics attainment in Class 2 was slightly higher, but there was a much broader spread of attainment levels in Class 1, which included very low-attaining pupils as well as a very high-attaining pupil (see Table 5).

Copies of the Scratch projects pupils created during the lesson were collected. After each lesson observation, audio-recorded semi-structured interviews with the teacher and a subset of pupils were conducted (three per class). The teachers selected at least one higher, one middle and one lower-attaining pupil (in mathematics) and a mix of genders. The pupil interviews were intended to identify what they learned during the lesson, what problems they experienced and the strategies used to overcome them. The teacher interviews were intended to establish the lesson learning aims, how they adapted the materials, their overall evaluation of the lesson and the challenges of teaching the lesson. This enabled the data sources to be

Table 5 Overview of School E classes

Class	No. of pupils (in observed lesson)	% of pupils achieving objective		
		Maths level	Objective 1 (algorithm)	Objective 2 (360°)
Class 1	26 pupils (2 pupils working in a pair – L1/L2)	Level 1 (1 pupil)	100%	100%
		Level 2 (8 pupils)	88%	38%
		Level 3 (9 pupils)	67%	22%
		Level 4 (6 pupils)	83%	67%
		Level 5 (1 pupil)	100%	100%
		Unknown (2 pupils)	100%	0%
		Total (26 pupils)	81%	41%
Class 2	29 pupils (4 pupils working in pairs – pair 1 = L3 and pair 2 = L3/L4)	Level 2 (2 pupils)	100%	100%
		Level 3 (20 pupils)	90%	60%
		Level 4 (5 pupils)	80%	80%
		Unknown level (2 pupils)	100%	100%
		Total (29 pupils)	90%	69%

triangulated during the analysis process. A deductive thematic analysis was undertaken on the field notes and interview transcripts using the 5Es framework to guide the coding process to identify explicit examples of their application during the lessons. A content analysis was then undertaken on the pupil projects to identify which learning objectives had been fulfilled and putative reasons for any project not fulfilling each of these objectives.

Summative Evaluation Findings

Below we describe the findings in terms of the teacher implementation and the pupil understanding of the research activity, which, to recap, had the following objectives: (1) to use the new pattern stamping algorithm; (2) to create a 360° pattern; (3) to use positive and negative numbers to move the sprite forwards and backwards. We then discuss the gap in each of the different cases between the intervention designers' intentions and teachers' implementation of the intervention and the implications this potentially may have had on the resulting pupil outcomes.

Teacher Data Teacher 1 spent a large part of the lesson with the pupils on the carpet at the front of the class, with approximately 15 minutes at the end of the lesson on the computers using Scratch. During the interview after the lesson the teacher stated how the key learning objective for her pupils was objective 3. She also explained how she typically structured a new activity into two lessons, with the first lesson introducing a new concept for the first time (which was the observed lesson) by letting pupils *explore* for themselves and show her what they can do - "*it's the one session where some children do show me an ability that I didn't assume them to have*". She then differentiated the activity during the second lesson to give the pupils who struggled a chance to repeat the activity and then to challenge those that excelled with some of the extension activities.

Whilst the children were on the carpet, Teacher 1 set pupils several tasks focused around *envisaging*. The teacher employed multiple tools such as showing the video of the two pattern-stamping algorithms multiple times with the pupils verbally saying the steps of each algorithm aloud. She also used physical walkthroughs of these algorithms, getting pupils to 'play the sprite' themselves as well as a pupil to direct her while the rest of the class observed. She used the video alongside the physical walkthrough, which helped highlight a mistake she made in one of the steps, allowing the pupils to recognise and debug this. She continued the use of envisaging when moving onto Scratch, making changes to her example scripts and asking pupils to predict the outcome and debug any errors. She used envisage tasks in combination with *exchanging*, with pupils discussing their ideas both in pairs and as a class, as well as encouraging them to *explain* their ideas or why they disagreed with another pupil's idea. The pupils were given the opportunity to *explore* later in the lesson when they moved onto Scratch. The teacher structured this exploration by showing them example patterns on the interactive whiteboard slides and also by displaying an example script on the interactive whiteboard. She then instructed them not to ask her further questions but to explore for themselves in Scratch. Towards the end of the lesson Teacher 1 made a *bridge* to mathematics once during the physical walkthrough task, asking pupils about the different types of angles they were turning (e.g. acute, obtuse, right angle). She did not explicitly discuss objective 2 during the class discussion about how to calculate the repeat and turn block values using their knowledge of a 360° total turn (although this had been introduced during previous activities).

Teacher 2 structured her lesson by alternating the pupils between sitting on the carpet at the front of the class and using the computers throughout the lesson, both beginning and ending the lesson on the carpet. During the interview after the lesson the teacher explained that although her initial learning objective for the lesson was objective 3 when she realized they had forgotten things they had covered in the previous lesson i.e. how to calculate the repeat and turn values using their knowledge of 360°, she changed the focus to revise this process. She explained that her class was much less diverse in terms of pupil attainment than Class 1 and so normally her pupils worked at a similar pace which was easier for her to manage, but during the observed lesson some pupils moved ahead (including a lower attaining pupil who was the first to complete a pattern script) whereas others took longer than she expected. Therefore she had to manage this difference on the fly by getting her teaching assistant to print out extension sheets to hand out to those that had finished.

Teacher 2 also used *envisaging* tasks: these were typically done within Scratch rather than physically as Teacher 1 had done. Teacher 2 often got pupils to predict the outcome of a script before she would run it, posing questions such as "*will it create a complete circle or a bit of circle?*" and getting the pupils to think about whether the order of the blocks will change the outcome. Furthermore she created scripts in Scratch with intentional errors that she would ask the pupils to identify and debug as a class. She also encouraged pupils to envisage how to create a script in Scratch from the given algorithm and used a physical stamp and inkpad to print on paper to support the visualization of the pattern creation algorithm. Teacher 2 included opportunities for pupils to *explain*, particularly during the early part of the lesson, asking them to explain different components of Scratch such as blocks and scripts to verify they understood later tasks she set them. She combined this with *exchanging* in a similar fashion to

Teacher 1, getting pupils to discuss with the person next to them before discussing as a class, giving all pupils the change to explain what they thought first.

After each carpet discussion Teacher 2 initially set the pupils a more open ended *exploratory* task to try and recreate some of the patterns they had looked at together, but then introduced additional structure after a short period of time for those pupils that needed further guidance. For instance, by demonstrating the strategies they needed to use to calculate values or pointing out hints written on the board about the steps their script should contain. She later increased this structure by going through an example solution as a class and then asking pupils to try and recreate this for themselves. Teacher 2 explicitly *bridged* to mathematics much more than Teacher 1, which could partly be due to the change in focus that she decided to make part way through the class. She dedicated time to working through the 360° total turn calculations and linked this with what they had previously covered in their mathematics lessons, and she also made links with the concept of multiples which they had also previously covered. Furthermore whilst the pupils were working on the computer she prompted them to "use their maths", asking a pupil to explain why this is important (i.e. so they can get their pattern to be a full circle).

Pupil Data The central finding, illustrated in Table 5, is that a higher percentage of pupils successfully achieved objective 2 in Class 2 than Class 1. In Class 2 the most common issue with the pupils' script was that the sprite turned more than 360° which would still produce a full circular pattern i.e. look visually correct (see Table 6), whereas in Class 1 although this was an issue many pupils also turned the sprite less than 360° which would create an incomplete circular pattern. Below we discuss our findings from the interviews with a subset of the pupils in relation to this objective.

Of the three case-study pupils in Class 1 who were interviewed, only the higher-attaining pupil achieved objective 2 (use mathematical knowledge of a 360° total turn to stamp a circular pattern). He chose for the last stamp to occur outside the repeat block so he could control the finishing position of the sprite, but he was able to still have his Tile sprite turning 360° in total. The middle-attaining pupil did achieve objective 1 (using new algorithm) and objective 3 (using positive and negative numbers to move forwards and backwards), but turned her Tile sprite too much. Within her script she chose to make the turn value and the move value the same number, which resulted in the Tile sprite turning more than 360°, although the resulting pattern would have looked correct (see Fig. 6). In the interview she explained how she had focused on ensuring her sprite was moving forwards and backwards, and how it was important to

Table 6 Issues identified in the projects of Class 1 (C1) and Class 2 (C2) pupils who had not achieved objective 2

Issue	Level 2		Level 3		Level 4		Unknown	
	C1	C2	C1	C2	C1	C2	C1	C2
Turns >360	1		3	5	1	1	1	
Turns <360	2		3	2			1	
Missing repeat/turn blocks			1	1	1			
Negative turn values (>360)	2							

use a negative number to move backwards. The lower-attaining pupil did not achieve any of the objectives. Her script indicated a misconception about the use of negative numbers within the pattern algorithm as she used a negative number within her turn block rather than the move block. Her script also outputted a pattern that looked incorrect as she only moved her sprite forwards (see Fig. 6).

In Class 2 both the higher and middle-attaining pupils achieved objective 2. The higher-attaining pupil was able to explain the values she had chosen to use within her script and also explained that it was important to use the same positive and negative numbers within her move block. The middle-attaining pupil explained "*I think I just guessed [the number] but the 30 I had done for the degrees because I was trying to do a full turn and this looks like a full turn so I think 12 × 30 is 360 so I put the 30 in there just to see what the difference is and this pattern came up*." She also highlighted that it was important to use maths skills within the activity, such as multiplication, to make the patterns. The lower-attaining pupil did not achieve objective 2, but did achieve objectives 1 and 3. Within her script she turned her Tile sprite more than 360°. She was also confused about the value she needed to turn her sprite in total explaining that she had chosen to turn her sprite 90° because she thought "*that is could be like adding to 160 so 90 add something to make 160*" (see Fig. 7).

Discussion and Concluding Remarks

Below we discuss our findings in terms of our initial research aims. We consider our first aim specifically in relation the concepts selected as a focus for this paper, that is: *What are the characteristics of an evidence-based curriculum intervention in both mathematics (e.g. 360° total turn) and programming (e.g. algorithm) for primary school pupils (aged 9–11 years)?* Our design research highlighted that the idea of algorithm is a particularly challenging concept for both pupils and teachers. Accordingly, this was a specific focus, which we introduced clearly and explicitly, and for which teachers were equipped with a range of supports to help them in communicating the concept to their pupils. Typically within primary computing lessons, algorithm is introduced through everyday or school activities, which only address limited characteristics of an algorithm such as order and have more than a little reliance on metaphorical examples such as making cups of tea or jam sandwiches (sic). Within our summative case study we saw both teachers use multiple ways to represent the concept of alternative algorithms, chiefly by using different forms of *envisaging* such as physical enactment or visualising the pattern that should be output on the screen.

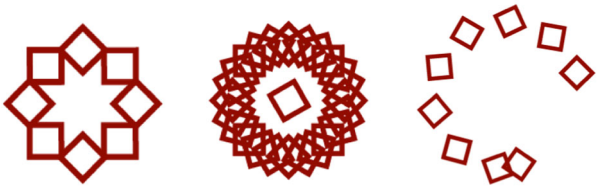

Fig. 6 Patterns produced by Class 1 pupils: Left = H; Centre = M; Right = L

Fig. 7 Patterns produced by Class 2 pupils: Left = H; Centre = M; Right = L

Using multiple forms simultaneously helped pupils as well as one of the teachers, to identify errors in their algorithm. Although both teachers employed a range of different strategies, both experienced high rates of success in terms of their pupils building scripts which followed the correct algorithm: this seemed to demonstrate the flexibility of the 5Es framework in guiding pedagogy to match different teaching styles and experiences as well as different ranges of pupil attainment. Perhaps unsurprisingly, Teacher 1 with less experience/confidence in Scratch chose to envisage much more using approaches away from Scratch such as the video or physical walkthroughs, whereas the more experienced and confident teacher was much more likely to use Scratch as one of her envisaging activities, that is to actually build or change the blocks in a script to show the different results, alongside acting it out. This appeared to be an effective strategy, the absence of which might explain any gap between design and implementation, a conjecture we will investigate further in relation to others concepts.

Although the data are far from conclusive, the relative success of Teacher 2 lends support to the conjecture that exploiting the affordances of the activities and programming solutions, adding another representation tends to engage more pupils and also is likely to lead to deep learning.

The 360° total turn concept was also challenging for both teachers and pupils. We noted the importance of returning to explicitly address this concept within off-computer activities and providing pointers towards key features such as expressing the number of times to repeat. Our main focus was to help pupils move beyond merely recognising that 360 is a 'special' number to a richer appreciation of why 360° is powerful in expressing the structure of the task. Our case study highlighted the need to return to this concept several times as although pupils had been introduced to it in earlier lessons many still were not able to apply this knowledge to their script. The more confident Teacher 2 was able to recognise this difficulty and as a result decided to adapt her lesson plan on the fly to focus more on bridging by providing pupils with a series of strategies, hints and a walk through of an example solution as well as making explicit links with previous related work they had undertaken in their mathematics lessons. It was clear that this approach helped many more pupils in Class 2 successfully to create 360° patterns. However, Teacher 1 preferred to give her pupils more time initially to explore each new concept to allow lower-attaining pupils (in particular) to demonstrate what they could do before then introducing increased structure in later lessons. Our findings highlighted that this approach did allow some lower-attaining pupils to achieve the learning objective with minimal support. This therefore allows the teacher during subsequent lessons to provide additional support to those pupils who actually require it rather than providing unnecessary support to those that are assumed to need it. Furthermore although we did not focus on pupil affect within this particular study, within the context of the constructionist foundations of the work it is worth mentioning that a high level of pupil engagement with the activity was observed within both classes, as relating learning to areas

of meaningful personal interest is important in the constructionist approach (Papert 1980, 1993). In the post-lesson interviews the teachers were universally positive about their pupil engagement with the activities, with Teacher 2 highlighting the benefit of this engagement for her pupils' mathematics learning "*It has been really good because they are really interested in the Scratch and it's good to put angles especially into a context that means something to them because otherwise it can be really bizarre really, intangible*".

The findings from our case studies suggest that our framework for action is flexible enough for teachers to adapt to their own teaching style and experience as well as the needs of their pupils to communicate key computational and mathematical ideas in different ways whilst staying true to the goal of the learning activity (we have observed no examples of what we might characterise as a 'lethal mutation' in the sense of Hung et al. (2010)). One limitation of our research is that it only provides a snapshot of the intervention, not allowing us to see how the gap is managed *over time*, how much teachers continue to employ the different aspects of the framework and if they have a tendency to focus more on specific Es or are able to adapt their use of the framework for different activities. In addition, we have yet to develop a model of how this impacts upon the pupils learning attainment over the course of the intervention and in other key important concepts, which build on these such as abstraction and regular polygons.

We now consider our second research aim - *What is the gap between the intervention designers' intentions and the teachers' implementation of the intervention in practice for specific computational and mathematical concepts and how does this impact the pupil attainment?* We have begun to answer this question in the case study, in which our design 'principles' of the 5Es are useful in analysing what teachers actually did in relation to what we hoped they would do. Our case study raised a number of relevant considerations in terms of defining the gap which include:

- Teacher confidence and subsequent use of the technology within their teaching to support the learning of both computational and mathematical concepts
- Teachers potentially increasing their emphasis on either the computational or mathematical-related learning aim
- The differing gaps between pupil attainment within a class and the teacher management of this difference
- Differences in selected pedagogical strategies employed to deliver the content, which may be influenced, for example, by confidence, teaching style or existing school practices.

Furthermore it is possible that some of the gaps between the original intentions and the resulting implementation may be influenced by the design of the initial PD sessions. Although specific examples of the 5Es were provided during these sessions, they are many more strategies that teachers may select beyond these e.g. for *envisaging* teachers could employ different modes of algorithmic expression often used with younger children such as music, dance or bee-bots/pro-bots. One option could be to provide space within the PD for teachers to share their own strategies with one another, which could feed into the redesign of the PD.

Our overarching theme in this paper has been to investigate the effectiveness of an intervention in programming and mathematics in terms of its the pedagogical and curriculum framework but also in terms of pupil learning of two key concepts. It is

worth stressing that while the choice of case study as a methodological heuristic is far from novel, the ambition of the research is surprisingly broad. We are asking whether *teaching* judiciously chosen mathematical ideas through Scratch programming in carefully designed ways is associated with children *learning* how better to express and reason mathematically. Furthermore what might be the conceptual and pedagogical obstacles and how might they be addressed?

Our ambition has resulted in what is essentially a set of design criteria, mapping (in a highly non-trivial way) between computational and mathematical. And we have constructed a framework for action which, by explicitly focusing on mathematics (most research involving programming is actually *about* programming), may help more effectively understand the relationship between computational and mathematical thinking, and thus design curricula (in both formal and informal senses) that are more successful than those we have now.

Our next step is to build on these preliminary findings and seek to establish how these outcomes may evolve over the course of the 2-year intervention.

Acknowledgements We would like to thank the Education Endowment Foundation for funding this work. We also thank all the teachers and students from our "design" and "trial" schools for their invaluable contributions to the design, development and evaluation of the SM intervention. Lastly we thank the reviewers for their valuable comments and suggestions, which have helped to further shape this paper.

References

Benton, L., Hoyles, C., Noss, R., & Kalas, I. (2016). *Building mathematical knowledge with programming: insights from the ScratchMaths project*. In: Proceedings of Constructionism 2016, pp. 25-32.

Brennan, K. (2015). Beyond technocentrism: supporting constructionism in the classroom. *Constructivist Foundations, 10*, 289–296.

Brown, N., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: the resurgence of computer science in UK schools. *ACM Transactions on Computing Education*, *14*.

Clements, D. H. (1985). Research on Logo in education: is the turtle slow but steady, or not even in the race? *Computers in the Schools, 2*, 55–71.

Clements, D. H. (1999). The future of educational computing research: the case of computer programming. *Information Technology in Childhood Education Annual, 1*, 147–179.

Clements, D. H., & Sarama, J. (1997). Research on Logo: a decade of progress. *Computers in the Schools, 14*, 9–46.

diSessa, A. A., & Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *The Journal of the Learning Sciences, 13*, 77–103.

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn coding? Proceedings of the 9th Workshop in Primary and Secondary Computing Education, pp. 60–69.

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1969). Programming Languages as a Conceptual Framework for Teaching Mathematics. Final Report on the First Fifteen Months of the LOGO Project.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments, 1*, 1–32.

Hatfield, L., & Kieren, T. E. (1972). Computer assisted problem solving in school mathematics. *Journal for Research in Mathematics Education, 3*, 99–112.

Hoyles, C., & Noss, R. (1992). *Learning mathematics and Logo*. Cambridge: MIT Press.

Hung, D., Lim, K., & Huang, D. (2010). Extending and scaling technology-based innovations through research: The case of Singapore. Inspired by Technology, Driven by Pedagogy: A Systematic Approach to Technology-Based School Innovations. Paris, OECD Publishing, pp. 89–102.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: a cross-case qualitative analysis. *Computers & Education, 82*, 263–279.

Lewis, C.M. (2010). How programming environment shapes perception, learning and goals: Logo vs. scratch. Proceedings of the 41st ACM technical symposium on Computer science education, ACM.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: what is next for K-12? *Computers in Human Behavior, 41*, 51–61.

Manches, A., & Plowman, L. (2015). Computing education in children's early years: A call for debate. *British Journal of Educational Technology.*

McCoy, L. P. (1996). Computer-based mathematics learning. *Journal of Research on Computing in Education, 28*, 438–460.

Miller, R. B., Kelly, G. N., & Kelly, J. T. (1988). Effects of Logo computer programming experience on problem solving and spatial relations ability. *Contemporary Educational Psychology, 13*, 348–357.

Nieveen, N., & Folmer, E. (2013). Formative evaluation in educational design research. Educational design research. T. Plomp and N. Nieveen, SLO, pp. 152–169.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: learning cultures and computers*. London: Kluwer.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books, Inc.

Papert, S. (1993). *Mindstorms: Children, computers and powerful ideas*. New York: Basic Books.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism, 36*, 1–11.

Passey, D. (2016). Computer science (CS) in the compulsory education curriculum: implications for future research. *Education and Information Technologies*, 1–23.

Prediger, S., Gravemeijer, K., & Confrey, J. (2015). Design research with a focus on learning processes: an overview on achievements and challenges. *ZDM, 47*, 877–891.

Resnick, M., Maloney, J., Monroy-Hernández, A. R., Eastmond, N. E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*, 60–67.

Salomon, G., & Perkins, D. N. (1989). Rocky roads to transfer: rethinking mechanism of a neglected phenomenon. *Educational Psychologist, 24*, 113–142.

Sentance, S., & Csizmadia, A. (2015). Teachers' perspectives on successful strategies for teaching computing in school. IFIP TCS 2015.

Sentance, S., &Csizmadia, A. (2016). Computing in the curriculum: challenges and strategies from a teacher's perspective. *Education and Information Technologies,* 1–27.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: towards an agenda for research and practice. *Education and Information Technologies, 20*, 715–728.

Watt, S. (1998). Syntonicity and the psychology of programming. Proceedings of 10th Annual Meeting of the Psychology of Programming Interest Group, pp. 75–86.

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. Proceedings of the 14th International Conference on Interaction Design and Children, ACM, pp. 199–208.

Yelland, N. (1995). Mindstorms or a storm in a teacup? A review of research with Logo. *International Journal of Mathematical Education in Science and Technology, 26*, 853–869.